1. What are prerequisites to use HiAi?
   1. To use HiAi, the android code that wishes to support it must create a jni folder in the main folder of the source code. Inside the folder, they must include the following files; Android.mk, Application.mk, classify\_async\_jni.cpp, classify\_sync\_jni.cpp, HiAiAippPara.h, HiAiModelManagerService.h, HiAiModelManagerType.h. Some of these files can be found in the DDK but it will be missing classify\_async\_jni.cpp and classify\_sync\_jni.cpp. It may be that these two files are to be created by the people using HiAi in their application, but a workable version is present in the Demo\_Source\_Code folder in the jni folder.
   2. You also have to include a several files depending on if the device being used is arm64-v8a or armeabi-v7a
      1. arm64-v8a requires the lib files found in lib64
      2. armeabi-v7a requires the lib files in lib
      3. Both of the folders are in the ai\_ddk\_lib
   3. The build.gradle on the app level also has to be modified;
      1. Under defaultConfig, you must include:

externalNativeBuild **{** cmake **{** cppFlags "-std=c++11 -frtti -fexceptions"  
 **}** ndk **{** abiFilters 'arm64-v8a', 'armeabi-v7a'  
 **}  
}**

* + 1. Then under android, include:

externalNativeBuild**{** ndkBuild**{** path "src/main/jni/Android.mk"  
 **}  
}**

* + 1. And

sourceSets **{** main **{** jni.srcDirs = []  
 jniLibs.srcDirs = ['src/main/libs']  
 **}  
}**

1. What are the detail steps to covert your model to HiAI enabled model?
   1. To convert a tensorflow .pb file into the HiAi enabled .om model, you must use the following command;
      1. ./omg --model name.pb --framework 3 --output ./modelname --input\_shape "input\_node\_name:n,h,w,c" --out\_nodes "node\_name1:0"
      2. --model is the name of the pb file
      3. --framework 3 is specifying that the model is tensorflow
      4. --output is the name of the resulting model, you do not need to append .om to this
      5. --input\_shape is the input node and the shape of its input before conversion. This can be in nhwc or nchw, depending on what the original .pb file required. If there are other inputs, place it in the quotations with a semicolon in between. IE: “input\_name1:n1,c1,h1,w1;input\_name2:n2,c2,h2,w2”
      6. --out\_nodes is the output node name before conversion. IE: node\_name1:0;node\_name1:1;node\_name2:0. Outputs of the same node are numbered from 0 in ascending order. For example, the second output is numbered 1.
2. What are the detail steps to use your converted HiAI model?
   1. To use the converted HiAi .om model, it is placed within the assets folder along with its corresponding labels.txt file. Load the paths into the application and pass them to the cpp functions.
   2. The cpp files require the image to be in a byte[] format for it to process as well, the .om file requires nchw for the image, so make sure the preprocessing takes care of that.
3. How to verify your HiAI enabled model?
   1. If the .pb model cannot be converted, it will automatically fail upon creation
   2. If the HiAi enabled model is created, it should work. To check, make a basic program with the above steps and compare it to the expected results.
4. What are the 6 test cases to present to CP in the future?
   1. Original model performance on cpu and gpu (And npu if applicable) (3 cases)
   2. Converted model performance on npu (there is also a cpu version, but we’re currently more interested in the npu) (1 case)
5. Presentation Content:
   1. Why are we doing this
      1. Make it clear, articulated, first page powerpoint
   2. What did we do to achieve this goal
      1. How did we approach it
      2. Methodology
         1. High level, so that others that don’t know about programming can do this/understand
         2. Audience: device lab
         3. How we did it, make it understandable
   3. What are the test cases
      1. Results
   4. Analysis
      1. Insight of the output
      2. What we learned
   5. How is this going to help CPs?